ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Laboratorio di
Sicurezza Informatica

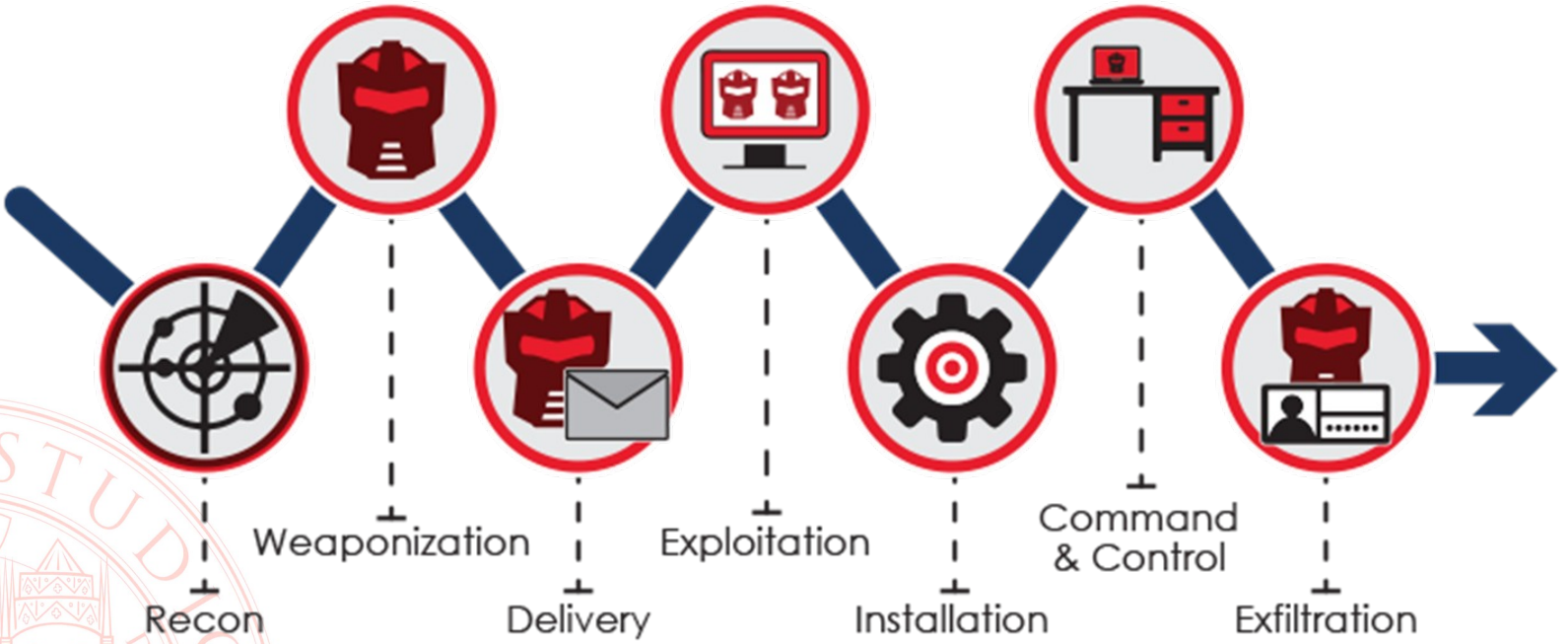
Monitoraggio

Marco Prandini

Dipartimento di Informatica – Scienza e Ingegneria

Finalità

- Le fasi di un attacco possono lasciare tracce.
- Individuarle con accuratezza e tempestività è di fondamentale importanza per evitare o limitare danni.



I termini del monitoraggio

- IDS = Intrusion Detection System
  - è genericamente un sistema in grado di rilevare tentativi di attacco
    - *signature based* (riconosce attacchi noti)
    - *anomaly detection* (riconosce deviazioni dall'uso standard)
- IPS = Intrusion Prevention System
  - semplificando: un IDS in grado di interagire con sistemi di controllo dell'accesso per bloccare il traffico malevolo
- SIEM = Security Information and Event Management
  - piattaforma che integra strumenti, politiche e procedure per la gestione integrata delle fonti di informazione e degli incidenti
- Parametri di qualità del rilevamento degli eventi
  - **Falso positivo (FP)**: segnalazione di attacco errata da evento innocuo
  - **Falso negativo (FN)**: attacco reale che non genera una segnalazione

NIDS, HIDS, EDR

- Due strategie di rilevazione
  - Basate sulla rete
    - NIDS / Network-based IDS
  - Basate sull'endpoint
    - HIDS / Host-based IDS
    - EDR / Endpoint Detection and Response
- NIDS usa i dati intercettati sui canali di comunicazione
  - sistema dedicato sul perimetro o con sonde, per il traffico di tutti i sistemi
- HIDS è un processo userland sul sistema da proteggere
  - vede il traffico diretto a un singolo sistema
  - monitora il filesystem, i processi, le attività utente
  - esamina in tempo reale i log file
  - verifica periodicamente contenuti e metadati dei file
- EDR può essere definito come un HIDS fortemente integrato col sistema operativo

NIDS

- Vantaggi
  - Visibilità di tutto il traffico, entrante e uscente
  - Richiede un solo punto di installazione
    - vero solo se rete semplice (dispositivi mobili che lasciano la rete??)
    - con più sonde: possibilità di ragionare su flussi
  - Un malfunzionamento non incide sugli endpoint
- Svantaggi
  - Maggior tasso di FP
    - processi legittimi possono generare occasionalmente traffico anomalo
  - Più soggetto a sovraccarico o evasione
    - es. pacchetti frammentati
  - Non può esaminare il traffico cifrato
  - Un punto di analisi per un'intera rete → richieste hardware
    - < 20-30 Mbps: raspberry PI
    - < 200-300 Mbps: pc desktop di un paio d'anni
    - < 1 Gbps: server almeno 8 core / 16 thread
    - > 10 Gbps: 40+ core e probabilmente serve accelerazione hardware

HIDS

- Vantaggi:
  - Minor tasso di FP
    - Pacchetti di rete sospetti possono essere correttamente classificati solo esaminando l'interazione con l'obiettivo finale
  - Economico
    - Sfrutta per definizione i sistemi già esistenti
    - Non molto impegnativo computazionalmente (distribuito)
- Svantaggi
  - Punti ciechi
    - Se un evento/pacchetti non lascia tracce sul filesystem è invisibile
    - Non valuta il traffico uscente (egress) – solo entrante (ingress)
    - Non individua scansioni che non toccano servizi attivi
  - Richiede l'installazione di un agente sulla macchina
  - Se la macchina è compromessa può essere neutralizzato

EDR

- Vantaggi su HIDS
  - in grado di raccogliere eventi dai device driver di filesystem e di rete e comunicazioni interprocesso
  - in grado di analizzare eseguibili e librerie al caricamento e a run time (system call, fork)
  - capacità anti-tampering
  - maggiori possibilità di risposta (isolamento di comunicazioni e di processi)
- Svantaggi
  - richiede interfacciamento stretto con OS (non così ovvio)
    - hooking
    - minifilters
  - difficile trovare soluzioni open e non molto costose

Host IDS – integrity check

- La rilevazione di intrusioni sull'host è tipicamente svolta per mezzo di un *integrity checker*
- Principio:
  - Si memorizza in un database lo stato del filesystem quando è certamente "pulito"
  - Si confronta periodicamente il filesystem col database
- Tra i più diffusi:
  - Tripwire (commerciale)
  - AIDE (fork FOSS di Tripwire)
  - AFICK

  https://www.sans.org/reading-room/whitepapers/detection/ids-file-integrity-checking-35327

Integrity checking – homemade

- I tool crittografici di base permettono di costruire a mano elenchi con hash dei contenuti dei file essenziali
- E i metadati? E i file speciali?
- In alcuni casi già la distribuzione del sistema aiuta
  - es. Linux RPM-based
    - file Size
    - Mode (include permessi e file type)
    - MD5 sum
    - Device major/minor number
    - readlink(2) path
    - User/Group ownership
    - mTime
- Dove si mette il database? È protetto da manipolazioni?

Integrity checkers

- Caratteristiche da valutare:
  - Algoritmi usati per calcolare le impronte dei file
  - Performance e dimensioni del DB
  - Capacità di proteggere i propri stessi binari
  - Capacità di proteggere il database
  - Portabilità
  - Complessità degli aggiornamenti
    - Del sw
    - Del database

AIDE

- AIDE è un controllore di integrità configurabile
- Il file `/etc/aide.conf` definisce i tipi di controlli da applicare a file e directory
- Processo
  - Un database di riferimento deve essere costruito su di un sistema *pulito*
  - Una scansione periodica confronta i file di sistema con il database in base alla configurazione e riporta le differenze rilevanti

https://aide.github.io/

http://doc.opensuse.org/products/draft/SLES/SLES-security_sd_draft/cha.aide.html

AIDE – tipi di controllo implementati

DIRECTIVE	DESCRIPTION
p	permissions
i	inode
n	number of links
u	user
g	group
s	size
b	block count
m	Mtime
a	Atime
c	Ctime
S	check for growing size
md5	`md5` checksum
sha1	`sha1` checksum
rmd160	`rmd160` checksum
tiger	`tiger` checksum
R	p+i+n+u+g+s+m+c+md5
L	p+i+n+u+g
E	Empty group
>	Growing logfile `p+u+g+i+n+S`

AIDE – qualche esempio

- La seguente riga di selezione esaminerà tutto nella directory /etc, esaminando in particolare il numero di collegamenti, l'utente che possiede un dato file, il gruppo che possiede un dato file e la dimensione del file:

```
/etc n+u+g+s
```

- Gli oggetti possono essere ignorati o saltati utilizzando un punto esclamativo (!), come nell'esempio seguente, che fa sì che AIDE ignori tutto in /var/log:

```
!/var/log/.*
```

- I pattern sono sottostringhe ancorate alla radice: attenzione alle esclusioni apparentemente specifiche!

```
!/var/log/maillog
```

- quello che si voleva dire, forse era:

```
!/var/log/maillog$
```

AIDE – qualche esempio

- Un esempio di configurazione

```
MyRule = p+i+n+u+g+s+b+m+c+md5+sha1
/etc p+i+u+g     # check only permissions, inode, user and group for etc
/bin MyRule      # apply the custom rule to the files in bin
/sbin MyRule     # apply the same custom rule to the files in sbin
!/var/log/.*     # ignore the log dir it changes too often
```

- Caratteristiche
  - Compilato, molto veloce
  - Integrabile con permessi estesi (acl, selinux)

AIDE – quick start

- Configurare
  - le regole di controllo
  - il nome del database di riferimento (usato per i controlli)
    - `database=file:/usr/local/aide/aide.db`
  - il nome del nuovo database prodotto ad ogni aggiornamento
    - `database_out=file:/usr/local/aide/aide.db.new`
- Inizializzare il database
  - `aide --init`
- Rinominarlo per usarlo come riferimento per i controlli futuri:
  - `mv /usr/local/aide/aide.db.new /usr/local/aide/aide.db`
- Lanciare un integrity check:
  - `aide --check`

AIDE – uso appropriato

- A seconda del caso, AIDE può essere
  - eseguito come strumento forense, solo se si sospetta / si verifica un'irruzione
  - programmato per segnalare regolarmente qualsiasi cambiamento interessante
    - Due problemi:
      - reimpostare periodicamente il database per interrompere la segnalazione di modifiche non dannose ai file
      - **difendere l'integrità del binario e del database di AIDE!**
- Per ridurre il rischio di eseguire un AIDE compromesso / su un DB compromesso:
  - utilizzare le firme HMAC per il file di configurazione e il DB
  - masterizzare il DB su di un supporto non riscrivibile
  - eseguire il software da un sistema diverso e affidabile
  - oppure, se non se ne dispone, da un supporto di ripristino
    - **fermo macchina!**

AFICK

- Configurazione molto simile ad AIDE ma con un numero minore di check supportati
- Caratteristiche
  - Può essere eseguito da media ottici read only per garantire la sua stessa integrità
  - Report CSV semplici e facili da importare in altri sw
  - Scritto in Perl, molto portabile

http://afick.sourceforge.net/

Log di sistema

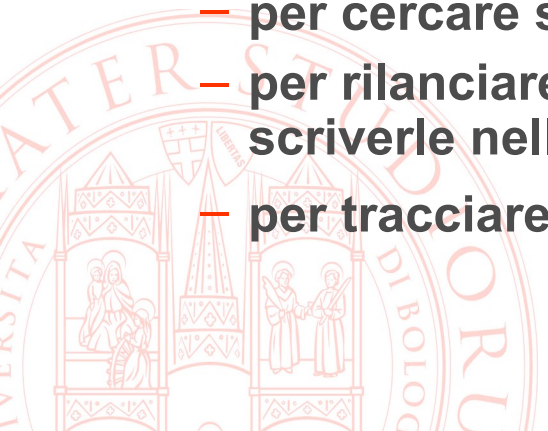
- I log (diari) tenuti dal sistema sono indispensabili per la diagnostica in generale, e in particolare per rilevare attività malevole o sospette
  - da processi utente
  - da processi kernel
- La loro stessa sicurezza va garantita, o l'attaccante semplicemente cancellerà le proprie tracce
  - Usare appropriatamente un integrity checker
  - Replicarli su macchine remote
- Logging su server remoto
  - Vantaggio aggiuntivo: centralizzazione
  - Implementazioni avanzate: shadow loggers
  - Problema: diventa un bersaglio appetibile
    - DoS

Linux logging

- Soluzioni comuni
  - Tipicamente producono file di testo
  - Nessuna garanzia di uniformità di formato a parte la marcatura temporale
  - BSD syslog (obsoleto)
    - klogd
  - Rsyslog
  - Syslog-ng
- In prospettiva integrato in systemd
  - *Journal*
  - Attivo dal boot, non dipende dall'avvio di altri servizi
  - Formato binario, visualizzabile con `journalctl`

In-kernel auditing

- Linux (≥4.18) supporta il tracciamento di ogni evento legato alle system call
- Il kernel invia messaggi a un demone user-space (`auditd`) secondo regole di configurazione
  - lette automaticamente da `/etc/audit/audit.rules`
  - o aggiunte a run time con `auditctl`
  - la destinazione predefinita è `/var/log/audit/audit.log`
    - gestito indipendentemente da rsyslog
- Sono disponibili strumenti
  - per generare report delle attività sul sistema (`aureport`)
  - per cercare specifici eventi (`ausearch`)
  - per rilanciare le notifiche di eventi ad altre applicazioni invece di scriverle nell'audit log (`auditspd`)
  - per tracciare un processo, analogamente a strace (`autrace`)



In-kernel auditing - esempi

dalla breve ma chiara guida su https://wiki.archlinux.org/index.php/Audit_framework

- `auditctl -w /etc/passwd -p rwxa -k KEY_pwd`
  - attiva un osservatore (watch) su /etc/passwd
  - lo fa scattare per ogni system call che tenti di eseguire read, write, execute, o attribute_change sul file
  - contrassegna le righe di log col tag KEY_pwd
- `ausearch -k KEY_pwd`
  - ricerca nel log le righe che hanno il tag specificato
- `auditctl -a exit,always -S chmod`
  - genera sempre (always) un evento loggato quando si ritorna (exit) dall'invocazione di una syscall chmod

Una guida più completa da SUSE:

https://documentation.suse.com/sles/12-SP4/html/SLES-all/part-audit.html

Analisi e gestione dei log

- Non basta scrivere gli eventi da qualche parte
- Analisi
  - estrazione del significato dei messaggi
  - serie temporali
  - correlazione file multipli
  - reazione in tempo reale
- Gestione
  - spazio
  - archiviazione

"Vedi la foresta, ma anche gli alberi"
– splunk.com

Software basici di analisi dei log

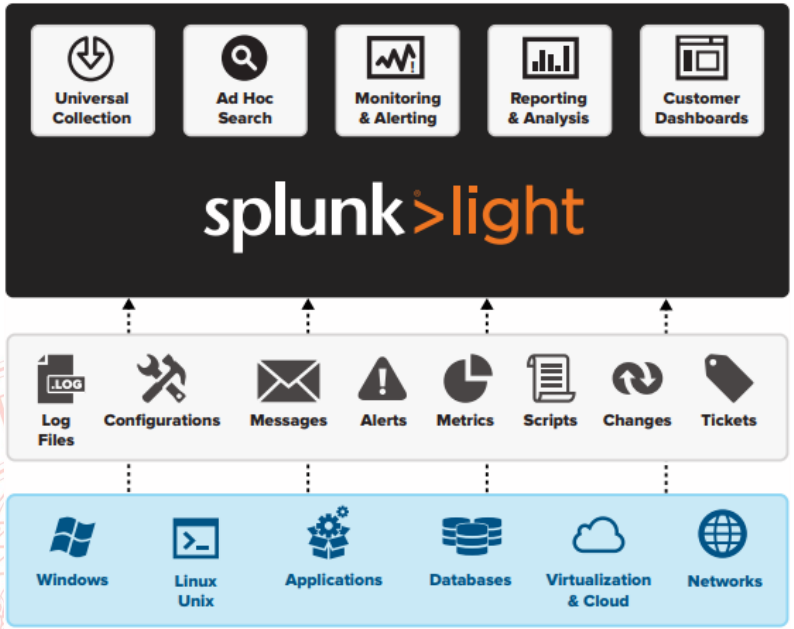
- Logwatch (System log analyzer and reporter):
  - https://sourceforge.net/projects/logwatch/
  - utile per l'analisi simultanea di diversi file
- Swatch (Simple WATCHer of Logfiles):
  - https://sourceforge.net/projects/swatch/
  - utile per la capacità di notificare in tempo reale la comparsa di righe corrispondenti a pattern dati
- Graylog2
  - https://www.graylog.org/products/open-source
  - strumento all-in-one, flessibile, di raccolta e ricerca

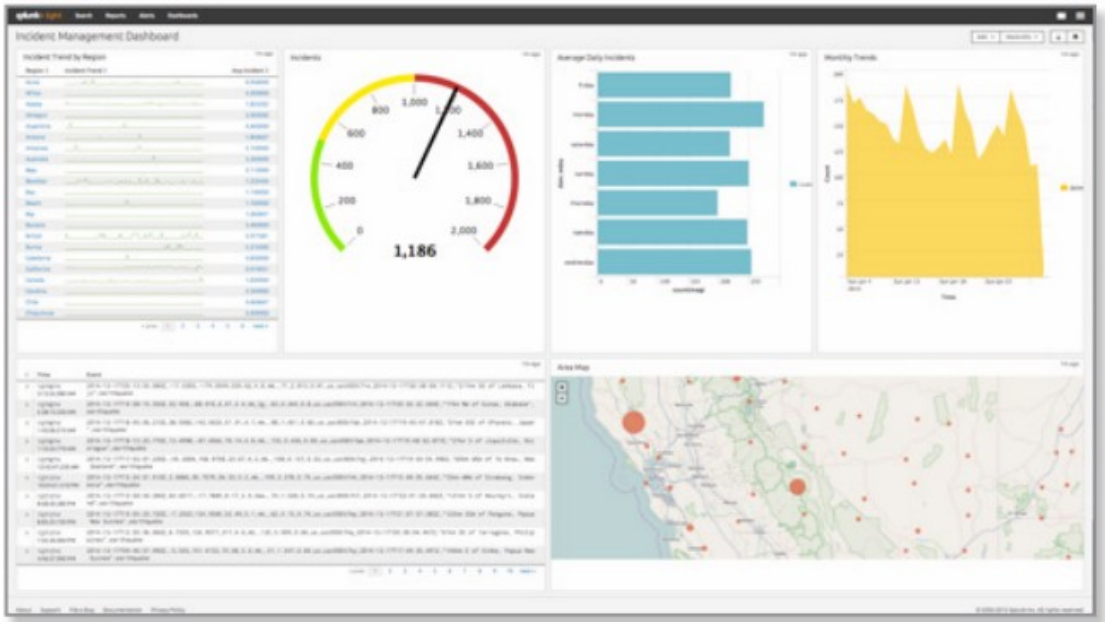
SIEM

- Aggregazione dei dati
  - raccoglie log / eventi da più fonti
  - normalizza e consolida i dati
  - sistema di interrogazione centralizzato
- Correlazione
  - collega eventi con attributi comuni in pacchetti significativi
  - genera automaticamente avvisi in base a condizioni specifiche
- Monitoraggio
  - grafici per visualizzare lo stato corrente
  - grafici relativi alla conformità
- Ritenzione
  - conservazione a lungo termine
  - analisi forense

SIEM

- Stanno evolvendo verso gestione integrata
  - qualsiasi tipo di "dato macchina" → **normalizzazione**
  - algoritmi di machine learning
  - reportistica avanzata
  - on premise o SaaS





Analisi dei dati di ispirazione cloud

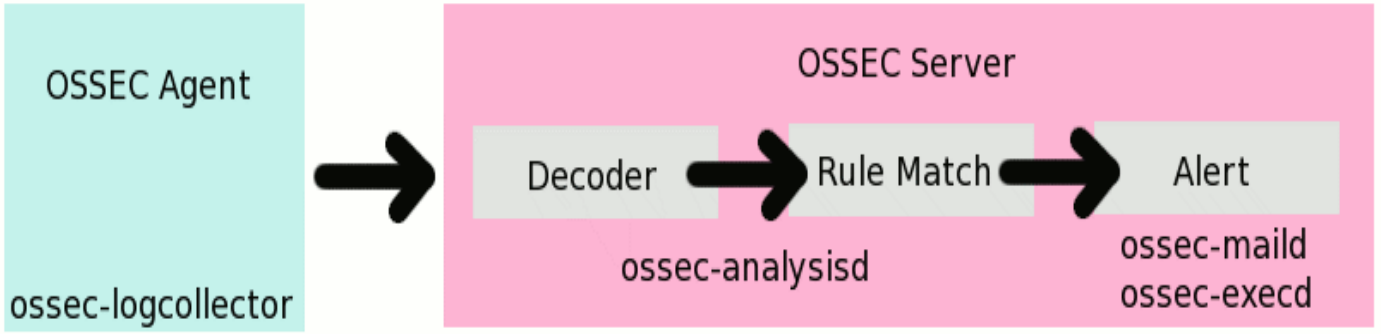
- Stack Elastic, Open Source
  - https://www.elastic.co/
  - Raccolta e esplorazione dei dati di log
  - definizione di un Elastic Common Schema per normalizzare
    - aperto, semplice, estendibile
    - limitato all'ambiente Elastic, verboso, non facile manutenzione delle customizzazioni
- Composto dai tre programmi:
  - Logstash: Pipeline di elaborazione dei log
  - Elasticsearch: Database Nosql
  - Kibana: Visualizzatore web-based per documenti in Elasticsearch

Un esempio di SIEM OSS: Wazuh

- Open source fork di OSSEC
  - per vedere rapidamente cosa è cambiato
    https://wazuh.com/migrating-from-ossec/
  - nel seguito si fa riferimento alle caratteristiche originali di OSSEC
- Funzionalità principali
  - Verifica della compliance a policy di sicurezza OpenSCAP
  - Raccolta dei log, analisi e conservazione centralizzata
    - Integrazione con Elastic stack
  - Filesystem integrity checking
  - Host-based IDS – non servono sonde in rete
    - monitoraggio del Windows registry
    - scansione per malware, rootkit, anomalie, processi offuscati, syscall inconsistenti, ...
  - Reazioni attive
    - built-in: RTBL (Real Time Black Listing)
    - qualsiasi cosa via scripting

OSSEC

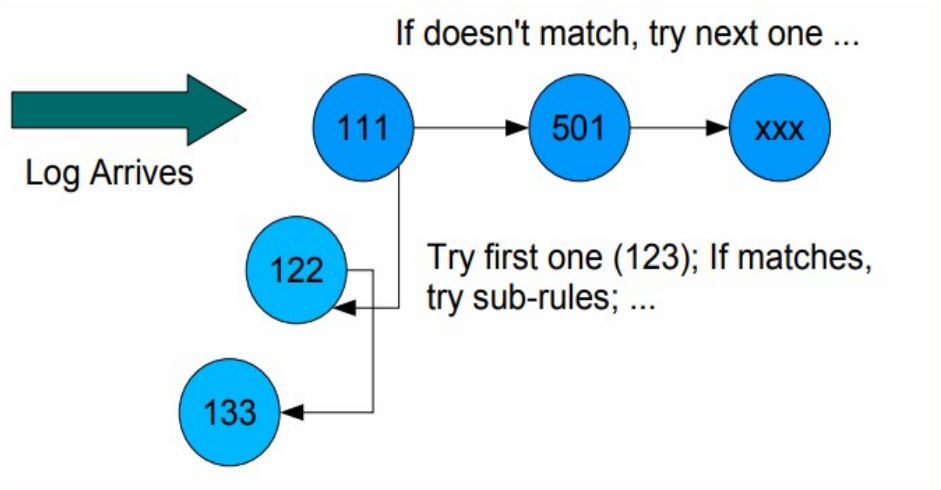
- Due modalità di funzionamento
  - locale, client-server
- Modalità client-server
  - i client ricevono la configurazione da un server
  - i client inviano i log al server su canale cifrato



- Comunicazione
  - standard syslog (UDP:514)
  - compressione
  - cifratura simmetrica (blowfish con chiavi scambiate manualmente)

OSSEC config / decoders e rules

- Parsing dei log file configurabile per mezzo di *decoders*
  - monitoraggio di file multipli
  - regole di parsing ed estrazione scritte in XML
  - forniscono i campi utili per l'attivazione delle *rules*
- Analisi dei dati per mezzo di *rules*
  - scritte in XML
  - componibili in gerarchia
    - livelli di priorità 1-15
  - ruleset pre-configurati per i servizi più diffusi



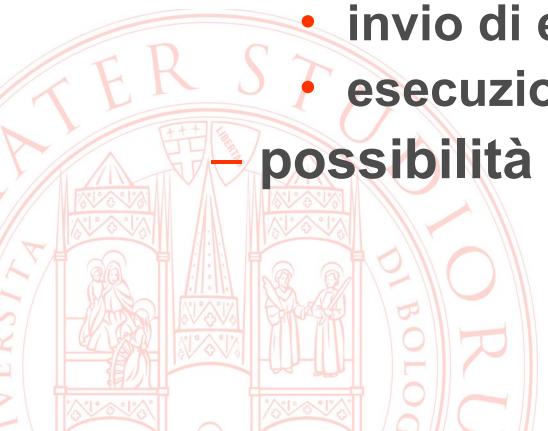
- Esempi

http://ossec-docs.readthedocs.io/en/latest/manual/rules-decoders/create-custom.html

https://sevenminuteserver.com/post/2010-09-25-writing-custom-ossec-rules/

OSSEC config / alerts

- Azioni predefinite – molte tra cui
  - Vari tipi di attacco ad applicazioni web
  - Attacco di forza bruta agli account via SSH
  - Buffer overflow e terminazioni anomale di processi
  - Eccezioni alle regole di controllo dell'accesso del traffico
  - Utilizzo di sudo
- Creazione di alert personalizzati
  - scattano in funzione delle *rules*
  - possono eseguire
    - logging dell'evento
    - invio di e-mail, sms, …
    - esecuzione di uno script
  - possibilità di **esecuzione su host multipli**

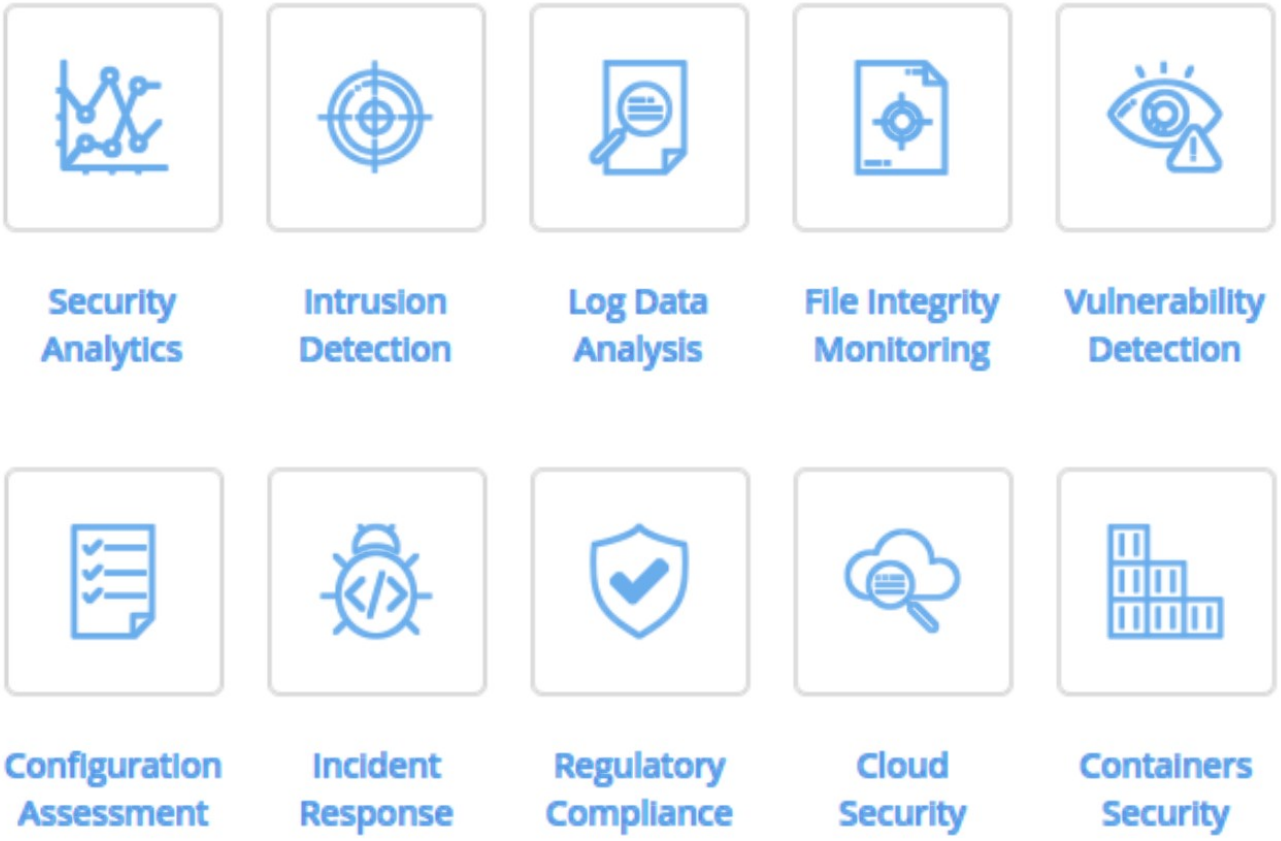


OSSEC funzioni avanzate

- Monitoraggio dell'output di script, ad esempio scan NMAP
  - alert quando un host sotto controllo cambia

  https://ossec.github.io/docs/manual/notes/nmap_correlation.html
- Reportistica
  - summary
  - database per successive elaborazioni
  - web UI (deprecata)
- https://ossec.github.io/

Wazuh functions



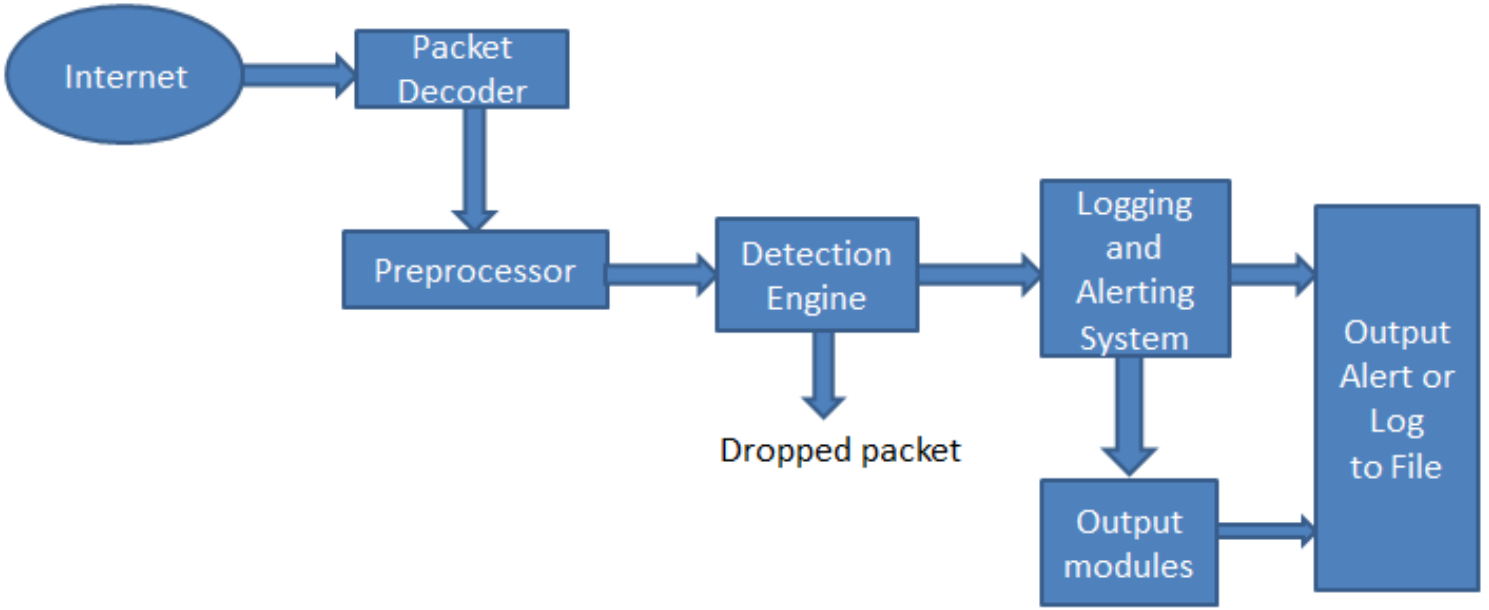
- https://wazuh.com/

Network IDS

- La rilevazione di attacchi che giungono via rete viene svolta analizzando il traffico in entrata/uscita
- Problema essenziale:
  - esaminare tutto il traffico senza rallentarlo
  - generando pochissimi falsi allarmi
  - senza lasciar sfuggire attacchi reali
- Due approcci:
  - Signature based: rileva flussi con caratteristiche notoriamente malevole
  - Anomaly based: rileva flussi che si discostano dalla “normalità”
- Tra i più diffusi
  - Snort
  - Suricata
  - Zeek (ex Bro)

SNORT

- http://www.snort.org/



SNORT

- Architettura di base:
  - Libpcap-based sniffing interface
    - cattura i pacchetti e li salva in un formato standard per l'analisi successiva
  - Rules-based detection engine
    - possibilità di utilizzare un vasto set di regole già pronte e di personalizzarle
  - Plug-in system
    - funzionamento estendibile aggiungendo moduli
- CAVEAT: Snort è uno strumento potente, ma per massimizzare la sua efficacia serve una competenza specifica ed un lungo affinamento della configurazione
  - stima dell'autore originale: 12 mesi di formazione per acquisire i fondamenti di intrusion detection, 24-36 mesi per diventare esperti

SNORT – Detection Engine

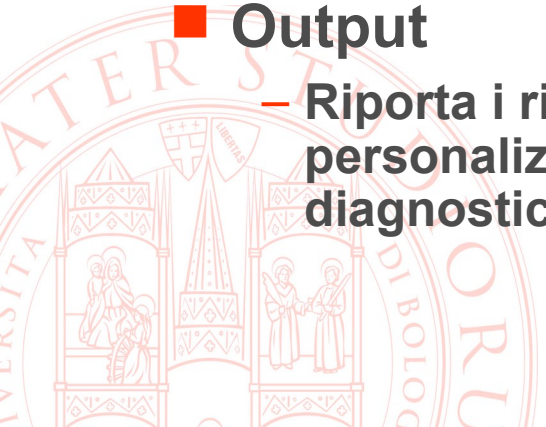
- Le regole definiscono le “signature” di un attacco, cioè l'insieme di caratteristiche per riconoscerlo
- Possono essere formate combinando più elementi semplici
- Possono riconoscere una molteplicità di scenari
  - Stealth scans, OS fingerprinting, buffer overflows, back doors, CGI exploits, ecc.
- Il sistema è molto flessibile e la creazione di nuove regole è relativamente semplice

```
alert tcp $EXTERNAL_NET 27374 -> $HOME_NET any (msg:"BACKDOOR
subseven 22"; flags: A+; content: "|0d0a5b52504c5d3030320d0a|";
reference:arachnids,485; reference:url,www.hackfix.org/subseven/;
sid:103; classtype:misc-activity; rev:4;)
```

SNORT – Plug-Ins

La struttura di base permette di attivare diversi moduli per le tre fasi principali

- Preprocessor
  - esamina e manipola i pacchetti prima di passarli al detection engine (evitando ad esempio la scansione di cose ovviamente innocue)
- Detection
  - ogni modulo implementa un singolo test semplice su di un singolo aspetto o parte di un pacchetto
  - può anche essere saltata se si vuole usare Snort solo per salvare traffico interessante da processare successivamente, fuori linea
- Output
  - Riporta i risultati degli altri plug-in (quindi consente la personalizzazione delle destinazioni e dei formati dei messaggi diagnostici)



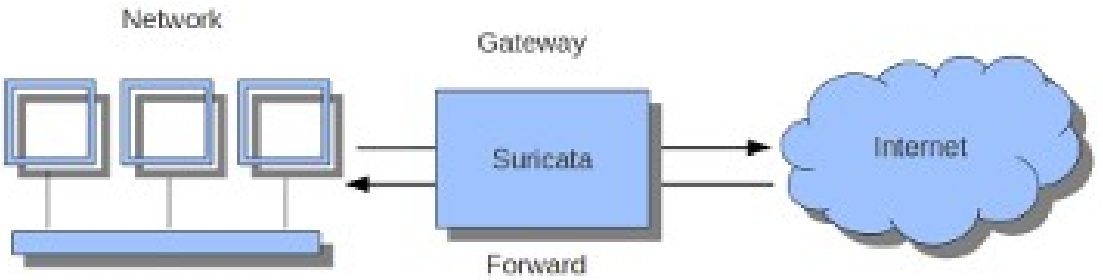
Suricata

https://suricata-ids.org/

- Configurazione:
  - Implementa un linguaggio per la rilevazione di signature
    - Compatibile con SNORT
  - Può essere configurato per rilevare anomalie
  - Può essere esteso con LUA per processing oltre la capacità del linguaggio a regole
- Caratteristiche di funzionamento particolari:
  - Riconosce automaticamente il tipo di traffico e adatta il dettaglio dei log
    - es. salva i certificati X.509 usati nelle connessioni TLS
    - salva l'header a livello applicazione per i protocolli più comuni
      - Deep Packet Inspection
- Output facilmente integrabile con molti strumenti di analisi e visualizzazione
- Molte fonti gratuite di signature
  - Emerging Threats
  - Talos
  - Positive Technology

Suricata

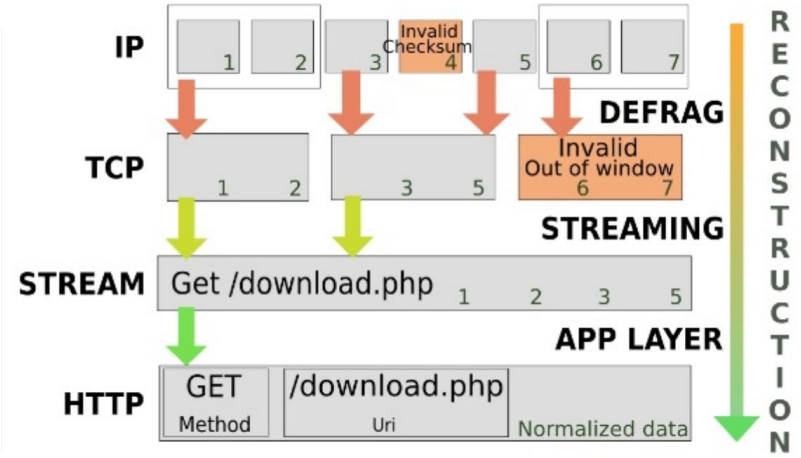
- Suricata può agire da IPS (Intrusion Prevention System)
  - Se interposto tra due reti, può non inoltrare il traffico malevolo

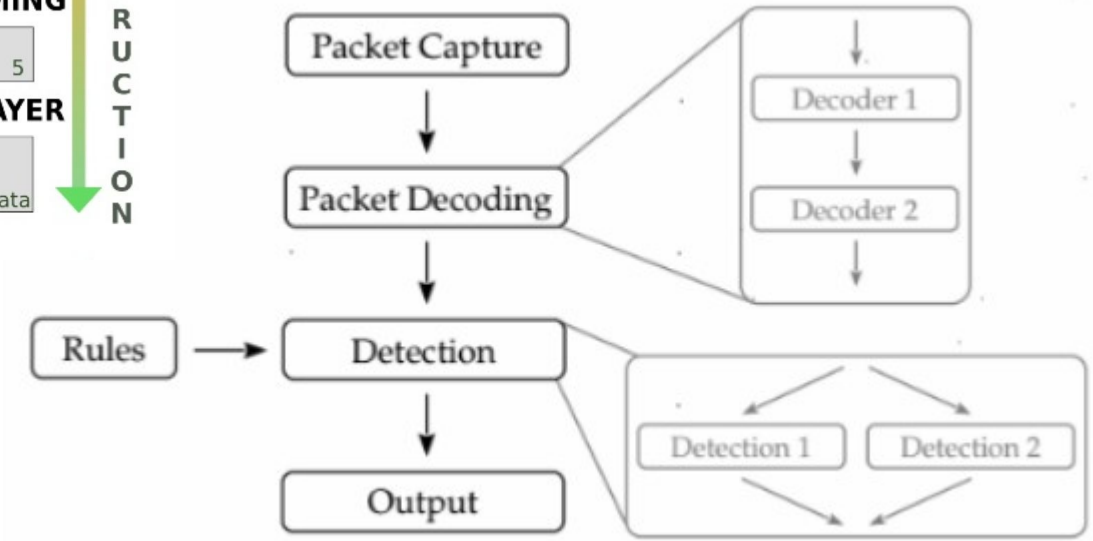


- Regole di pattern matching sul traffico
- Possibilità di operare "sul filo"
  - interfaccia in promiscuous mode
- Possibilità di operare "offline"
  - alimentato da file *pcap*
  - i file possono essere generati in tempo reale da probe integrate su apparati di rete o su host

Suricata

- Sistema modulare per ricostruzione, decodifica ed estrazione dei dati dai pacchetti e successiva classificazione





Zeek

https://zeek.org/

- Decodifica vari protocolli applicativi (HTTP, SSL, DNS, SMB e molti altri)
- Analizza TUTTO il traffico
  - Non utilizza le firme per individuare traffico malevolo
- Ha un motore di scripting molto potente:
  - per estrarre file interessanti
  - usa filtri di Bloom per cercare corrispondenze “intelligenti”
  - supporta la geolocalizzazione
  - blocco attivo della connessione chiamando API del firewall
  - calcolo dell'entropia su letture e scritture SMB per rilevare i ransomware mentre cifrano i file